

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
17 May 2001 (17.05.2001)

PCT

(10) International Publication Number  
**WO 01/35594 A2**

(51) International Patent Classification<sup>7</sup>: H04L 29/00

(21) International Application Number: PCT/GB00/04274

(22) International Filing Date:  
8 November 2000 (08.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
99308869.9 8 November 1999 (08.11.1999) EP  
0000465.5 10 January 2000 (10.01.2000) GB

(71) Applicant (for all designated States except US): **BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY** [GB/GB]; 81 Newgate Street, London EC1A 7AJ (GB).

(72) Inventor; and

(75) Inventor/Applicant (for US only): **GARYFALOS, Anargyros** [GR/GB]; 10 Hunters Ride, Martlesham Heath, Ipswich, Suffolk IP5 3SQ (GB).

(74) Agent: **ROBINSON, Simon, Benjamin**; BT Group Legal Services, Intellectual Property Department, Holborn Centre, 8th floor, 120 Holborn, London EC1N 2TE (GB).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published:**

— Without international search report and to be republished upon receipt of that report.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.



**WO 01/35594 A2**

(54) Title: **TELECOMMUNICATIONS CONTROL PROTOCOL PROCESSING**

(57) Abstract: A software application program for processing control messages constructed in accordance with a telecommunications control protocol, said program having an object-oriented structure and being arranged to dynamically load an object class, whereby a control message is processed, at run time.

## TELECOMMUNICATIONS CONTROL PROTOCOL PROCESSING

This invention relates to telecommunications control protocol processing, and in particular a software application program, and data processing means, for  
5 processing control messages constructed in accordance with a telecommunications control protocol. The invention is particularly, but not exclusively, directed to text-based application-layer control protocols.

The Session Initiation Protocol (SIP) is an application-layer control protocol for creating, modifying and terminating sessions having one or more  
10 participants. These sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution. Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these. SIP supports session descriptions that allow participants to agree on a set of compatible media types. It also supports user mobility by proxying and  
15 redirecting requests to the user's current location. SIP is not tied to any particular conference control protocol. There is widespread interest in the protocol, especially for telephony-related applications. SIP was proposed by the Internet Engineering Task Force (IETF) group and is now a proposed standard published as RFC 2543.

20 The entities used in SIP are user agents, proxy servers, redirect serves and location servers. A user is addressed using an email-like address "user@host", where "user" is a user name or phone number and "host" is a domain name or numerical Internet Protocol (IP) address. SIP defines a number of request types, in particular INVITE, ACK, BYE, OPTIONS, CANCEL, and  
25 REGISTER. Responses to SIP messages indicate success or failure, distinguished by status codes, 1xx (100 to 199) for progress updates, 2xx for success, 3xx for redirection, and higher numbers for failure. Each new SIP transaction has a unique call identifier (call ID), which identifies the session. If the session needs to be modified, e.g. for adding another media, the same call identifier is used as in  
30 the initial request, in order to indicate that this is a modification of an existing session.

The SIP user agent has two basic functions: listening for incoming SIP messages, and sending SIP messages upon user actions or incoming messages.

The SIP user agent typically also starts appropriate applications according to the session that has been established. The SIP proxy server relays SIP messages, so that it is possible to use a domain name to find a user, rather than knowing the IP address or name of the host. An SIP proxy can thereby also be used to hide the location of the user. A redirect server returns the location of the host rather than relaying the SIP message. Both redirect and proxy servers accept registrations from users, in which the current location of the user is given. The user's location can be stored at a dedicated location server. Typical signalling sequences between two user agents UA (A) and UA (B) over an IP network using an SIP redirect server (RS), and an SIP proxy server (PS), each also using an SIP location server (LS), are shown in Figs. 1 and 2, respectively.

SIP is typically implemented by transmitting Internet Protocol (IP) packets. SIP is independent of the packet layer and only requires an unreliable datagram service, as it provides its own reliability mechanism. While SIP typically is used over UDP or TCP, it could be used over frame relay, ATM AAL5 or X.25.

SIP is a text based protocol and is based to a certain extent (in terms of syntax) on the HTTP protocol. A typical messages consists of a single request line, a number of header lines and a message body.

The request line indicates the type of the messages, the message destination and the SIP version it complies with. The following is a typical example:

INVITE sip:Richard@bt.com SIP/2.0

A header line contains the name of the header type, followed by a semicolon and the contents as these are defined for the specific header. Consequently, each header type is used for a specific purpose (either to indicate some parameters or to issue a request). The following are typical examples:

From: sip:Richard@bt.com

To: sip:Steve@bt.com

Subject: Official meeting

The message body may be of any content, although it usually has contents formatted in accordance with the Session Description Protocol (SDP).

Extensibility of SIP may be achieved by the definition of new header types. For example, a desired charging scheme may be implemented using a new

header type which contains several parameters. The following is an example of an SIP header line including an example of a new header type (BT\_charge):

BT\_charge: Allowed\_bandwidth=X, Tariff=Y

5           The definition of such new header types is a mechanism which offers both flexibility and extensibility.

          There is a drawback with the definition of new header types in known implementations of SIP. For example, in a direct User Agent to user Agent interaction User Agent A may support the extension BT\_charge whereas User Agent B may not. In this example, BT\_charge may specify a number of new headers or even new request types that allow an efficient negotiation on charging issues about the multimedia session to be set up. When User Agent A sends an invitation to B, it may explicitly specify that it requires B to support the BT\_charge extension in order to proceed. This can be done by the inclusion of a standard SIP header called "Require" in the first INVITE request. In this way, when User Agent A sends an invitation to B containing the Require header, B determines that it does not support the BT\_charge extension and sends a negative response indicating the reason, that is to say, BT\_charge not supported. Interaction between A and B is then terminated. In order for B to be able to support the BT\_charge extension, a system administrator must shut down B obtain details of the required extension and then implement the new header and request types by loading a new program or modifying the system code, and restart the User Agent. The time and effort required to do this renders the process unviable, both in terms of cost and performance. In addition, further problems may arise due to differences in the implementations of the respective extensions.

          In accordance with one aspect of the present invention there is provided a software application program for processing control messages constructed in accordance with a telecommunications control protocol, said program having an object-oriented structure and being arranged to dynamically load an object class, whereby a control message is processed, at run time.

          By using the dynamic class loading capabilities of an object-oriented programming language, new classes can be loaded into an already running telecommunications control protocol implementing system and can be used as

though they were available at the time the system was started. Thus, control messages containing new, or modified, protocol elements can be handled by the system without the necessity for a new program to be loaded, or the system to be shut down for modifications to its program code to be made. In this way delay  
5 is reduced to an acceptable level.

One example of an object-oriented programming language having dynamic class loading capabilities is Java (Registered Trade Mark). When a Java Virtual Machine (JVM) is started up, the byte-code for various classes is loaded into the JVM prior to commencement of execution of the specified program. The classes  
10 loaded include all those that are needed for the specified program to run. Up to this point, the process of class loading is performed using native code which assumes that the byte code files for the required classes reside on the local file system, and may be located by making use of the CLASSPATH environment variable.

15 Once execution has commenced, it is possible to dynamically load further classes in one of two ways. If the byte code file of the required class is available locally, and can be located by following the CLASSPATH, then the static method `Class.forName ()`, which is briefly documented in Appendix A1 and Appendix A2 and, can be used to load the class as a system class. If the byte code file is not  
20 available locally, or cannot be traced via the CLASSPATH, then it can be loaded from a remote source by means of a Java Remote Method Invocation (RMI) `ClassLoader` function or a Java `URLClassLoader` function.

In accordance with a further aspect of the invention there is provided a software application program for processing control messages constructed in  
25 accordance with a telecommunications control protocol, said protocol defining a control message to have a construction including one or more protocol elements of a predefined format and having a plurality of respective types, wherein said program is arranged to process a control message by means of a different element type processing means for each of said plurality of respective types of at least  
30 one of said protocol elements, and, on receipt of a control message from a remote party that indicates that an element type processing means for a type of said at least one of said protocol elements is not available locally, to retrieve said

unavailable element type processing means remotely in order to process a control message received from, or to be sent to, said remote party.

The two parties involved can then conduct communications in accordance with the desired telecommunications control protocol, even where initially the two  
5 parties are running different versions of the protocol.

The program preferably has an object-oriented structure, in which the processing means is or are one or more object classes. The locally unavailable object class or classes are preferably retrieved by dynamically loading the missing class or classes during run time. As mentioned above, the RMI ClassLoader class  
10 provided in Java can be used to allow such dynamic class loading from a remote file system.

In a preferred embodiment, the locally unavailable object class or classes are retrieved from a source identified in the control message sent by the remote party.

15 Locally unavailable Java byte code files may be retrieved from a location identified in the control message and loaded using the method URLClassLoader documented in Appendix A3. In this way, control messages containing new, or modified, protocol element types can be handled in a seamless manner even if the respective object class or classes are locally unavailable and their remote locations  
20 initially unknown to the receiving party.

Thus, in embodiments of both aspects of the invention, the program is structured to allow not only new header types, but also new request types, to be processed using classes loaded by a dynamic loading function during run time.

Preferred embodiments of the invention will now be described, by way of  
25 example only, with reference to the accompanying drawings, in which:

Figures 1 and 2 show typical session control signalling used in SIP;

Figure 3 shows a schematic illustration of an SIP control program to be implemented on a user side device in accordance with an embodiment of the invention;

30 Figure 4 shows a schematic illustration of a part of the SIP control message handling method of the program illustrated in Figure 3;

Figure 5 shows a generic class diagram, using a Universal Modelling Language (UML) notation, of the classes in a SIP protocol program component according to an embodiment of the invention;

Figure 6 shows an event diagram at the arrival of an SIP message, using the SIP protocol component illustrated in Figure 5;

Figure 7 is an event diagram showing the parsing of an SIP message using the SIP protocol component illustrated in Figure 5; and

Figure 8 is an event diagram showing the construction of an SIP message using the SIP protocol component illustrated in Figure 5.

Referring now to Figure 3, in this embodiment of the invention an SIP control programme 2, implemented in Java and to be run on a user-side computer workstation with an IP connection, includes an SIP protocol component, ServicePack component 6 and a MultimediaApplication interface 8.

The ServicePack 6 is responsible for containing and operating any call control services: these are created by a service developer.

The MultimediaApplication interface 8 is a simple Java Interface with certain functions. It is intended to be inherited by the actual Multimedia application (i.e. videoconferencing, IP telephony tools, etc) in order to offer a standard API to the SIP control program 2.

The SIP protocol component 4 is responsible for handling all activities related to the protocol functionality (parsing/constructing messages, setting timeouts, etc). It "translates" them into events out of the context of SIP and notifies the other parts on the generation of these events. For example, when receiving an SIP request, it will send the appropriate temporary responses and then it will pass the relevant event type to the ServicePack 6. Subsequently, the defined call control will be triggered and the final response type will be passed back to the SIP protocol component 4. Based on that response, the final response will be sent back to the calling user agent and if necessary the object inheriting the MultimediaApplication interface will be notified (i.e. Start/End application, pass parameters, etc).

The main parts of the SIP protocol component 4 are the following:

An API part which offers a standard interface to the external applications;

A Headers part which contains header classes for all the different header types supported by the SIP protocol component;

A Message Handling part which contains message parser and constructor classes;

- 5       A User Agent part which includes classes forming the front end part of the application;
- and

A Handlers part which contains all the class objects (handlers) each of which is responsible for handling the operation of the protocol for different  
10    respective request types received in incoming messages.

The modularity of the Headers part and the Handlers part facilitates extensibility both in terms of changing a single header or handler class, and in terms of adding a new header or handler class. More explanation on this part is provided below.

- 15       Figure 4 shows an overview of the Message Handling part operating on an incoming SIP control message 10.

Whenever a packet containing an SIP control message 10 arrives, it is initially in the form of a text string containing the request line and the header lines. The message 10 is passed to an instance of a MessageParser class 12,  
20    which parses the text string and creates an instance of an SIPMessage object 14. This class contains a vector list 16 of different objects corresponding to the request line and all the header lines contained in the original message. The vector list 16 objects illustrated in the example of Figure 4 include a request line object 18, a VIA header object 20, a FROM header object 22, a TO header object, and  
25    one or more other header objects 26.

The remaining parts of the application can now operate on the SIPMessage object 14 and they retrieve the required information by questioning the corresponding object of the Vector list.

Whenever the MessageParser 12 reads a line from the textual  
30    representation of the message, it takes the first word (which denotes the type of the header), and by using the method Class.forName() (Appendices A1 and A2), dynamically searches for and loads a locally available header class with that name. If the header class is not found or the message contains an header that

identifies a remote location where the locally unavailable header class can be retrieved, then the method URLClassLoader (Appendix A3) is used to retrieve and dynamically load that header class.

For instance, using the Class.forName () method, if the MessageParser 12  
 5 reads the line "BT\_charge:Allowed\_Bandwidth=X Tarrif=Y", the MessageParser will dynamically search for the class type BT\_charge indicated at the start of the header line, and through a standard function (ReadContents in this Java embodiment) will pass the rest of the line to be parsed by and included in an instance of the BT\_charge header class. If the class type BT\_charge is not found  
 10 in local storage using the Class.forName () method the MessageParser will use the URLClassLoader method to first retrieve the class type BT\_charge from a remote location identified in the message. The location identifier may be a URL. In order to support this additional functionality a new header type is required. For instance, a new header type can be defined as:-

15

```
Class_Repository = ("Class-Repository" | "cl") ":" 1 * URL
```

For example:

```
Class_Repository = http://www.jungle.bt.co.uk/SIPheaders/
```

```
Class_Repository = http://www.jungle.bt.co.uk/SIPheaders/,
```

20

```
http://www.bt.com/SIPheaders
```

A new header of the above type may be used when an SIP User Agent indicates to another User Agent the location of one or more a header classes that are necessary for an SIP session to take place. For example, if User Agent A  
 25 sends an SIP invitation message to User Agent B, the message may include a Class-Repository header which identifies the location of one or more header classes. On receipt of the message User Agent B checks whether the header classes are available locally and loads any that are not from the URL location identified in the Class\_Repository header. Since the classes are loaded at run time  
 30 the interaction between A and B is seamless and continues as if all the header classes were available locally to B.

Although it may be argued that the above approach to parsing different elements of the incoming SIP message 10 imposes a performance penalty, it

introduces the advantage that message parsing is performed in a way which facilitates the extensibility of the application to handle new header types. Using this technique, to change the SIP protocol to support a non-standard extension, the following steps are carried out (exemplary code fragments are available from

5 Appendix B):

1. Create a header class that implements the Header interface (see Appendix B.1).

2. Implement the functions defined by the interface and those needed for the operation of the class (See Appendix B.2).

10 3. Store the new class in storage accessible to the SIP protocol component during run time. The storage may be either local or remote, with the MessageParser 12 dynamically loading the new class during run time.

The code which allows this flexibility resides in the MessageParser class (See appendix B.3). Since at least in Java the name of a class cannot contain any  
15 "-" characters, whenever the MessageParser 12 replaces it with the "\_" character instead (e.g. dynamically loading a class named call\_id instead of call-id).

Figure 5 is a generic class diagram, using UML notation, showing the most important functions, attributes and relations between the classes that implement the dynamic class loading feature of the SIP protocol component 4.  
20 This component 4 includes the MessageParser class 12 and the SIPMessage class 14 referred to above. Also included is a UserAgent class 30 which implements an SIP interface 32, whereby access to the SIP protocol component 4 by external applications is provided. The UserAgent class 30 contains a UDP class 36, whereby IP network functions (datagram packet receipt and sending) is enabled.  
25 The UserAgent 30 uses the MessageParser class 12 to create an instance of the SIPMessage class 14. The SIPMessage class 14 contains one or more header object classes 42. In Figure 5, the header object classes 42 are indicated generically, with individual classes corresponding to each currently supported SIP header type and possible future SIP header type extensions. Each example of the  
30 header objects class 42 implements a standard header interface 44, whereby the SIPMessage accesses the header object class 42.

The UserAgent class 30 also contains zero or more instances of a UAThread class 46. One instance of the UAThread class 46 is created for each

call ID currently handled by the SIP protocol component 4. The UAThread class 46 contains a handler object 48, which implements a handler interface 50 whereby the handler object is accessed by the UAThread class 46. In Figure 5, the handler object classes 48 are indicated generically, with individual classes 5 corresponding to each currently supported SIP request type and possible future SIP request type extensions. Each example of the handler objects class 48 implements a standard handler interface 50, whereby the UAThread class 46 accesses the handler object class 48. At any one time, an instance of the handler object class 48 uses either an incoming SIPMessage, an outgoing SIPMessage, or both. Therefore, the handler object class may use up to a maximum of two SIPMessages. In Figure 5, the indication "Handler Object" does not define the name of a class, but stands for any class that implements the handler interface. Apart from the inherited functions, each "Handler Object" may have different attributes and functions. The same applies to the indication 15 "Header Object".

The UserAgent class 30 also uses a MessageConstructor class 52, which reads from the SIPMessage class 14 in order to construct an SIP control message, formatted as a text string with the various request lines, message lines, header lines and message body lines as specified in the SIP protocol, to create an 20 outgoing packet to be sent by the UDP class 36.

Figure 6 illustrates an event sequence beginning with the receipt of an SIP control message by the SIP protocol component 4. Wherever appropriate, instances of classes described in Figure 5 are annotated with the same reference numerals, incremented by 100. In step 1, a UDP object 136 passes the received 25 text string message to a UserAgent object 30. In step 2, the UserAgent object 136 passes the text string message to a MessageParser object 112, which in step 3 creates an incoming SIPMessage object 114A. The creation of the incoming SIPMessage object is described in further detail below with reference to Figure 7. In steps 5 and 6, the SIPMessage object is passed back to the UserAgent object, 30 which then retrieves the caller ID from the caller ID header object in the SIPMessage object 114A. The UserAgent object then creates a UAThread object 146, having the appropriate caller ID as an attribute, using the incoming SIPMessage object 114A. The UAThread object 146 then retrieves the type of

SIPMessage from the SIPMessage object 114A in steps 8 and 9, and initiates a handler object 148 of the appropriate request type with the incoming SIPMessage object 114A, in step 10. In step 11, the handler object 148 processes the incoming SIPMessage object 114A and decides on a response thereto, in the form of a type of outgoing control message to be sent. In step 12, the handler object 148 creates an instance of the SIPMessage class 114 in the form of an outgoing SIPMessage object 114B, containing the appropriate request line and header objects as a vector list. In step 13, the handler object 148 passes the outgoing SIPMessage object 114B to the UAThread object 146, which in turn passes the outgoing SIPMessage object 114B to the UserAgent object 130, calling on the UserAgent object 130 to send the outgoing SIPMessage. In step 15, the UserAgent object 130 creates a MessageConstructor object 152, calling on the MessageConstructor object to create a text string message to be included in an outgoing UDP packet. The Message Constructor object 152 in steps 16 and 17 retrieves the appropriate header objects, and thereby, in step 18, constructs the text string to be sent out in the UDP packet which is then passed to the UserAgent 130 in step 19. Finally, in step 20, the UserAgent object 130 calls on the UDP object 136 to send the constructed SIP control message on an appropriate interface.

Referring now to Figure 7, when the MessageParser object 112 is requested by the UserAgent object 130 to create an SIP message, step 1, the packet is passed as a string object 110. The MessageParser calls on the object and reads the first line of the SIP protocol message, steps 2 and 3. In step 4, the MessageParser 112 identifies the type of request of the message, and constructs the appropriate incoming SIP message object 114A.

Next, the MessageParser object 112 reads the next line, the first header line, of the message string and dynamically loads the appropriate header class, step 8. It then passes the line string to the appropriate header object 126A, and adds the header object 126A to the incoming SIPMessage 114A. The operation is repeated, in steps 11 to 15, for the next line of the incoming SIPMessage, and for as many header lines as present in the incoming SIPMessage. Once the vector list of header objects is completed, the SIPMessage object 114A is passed back to the UserAgent 130.

Referring now to Figure 8, when the Message Constructor object 152 is requested to create an outgoing SIPMessage string by the UserAgent 130, step 1, the Message Constructor object 152 first reads the type of SIP request to be constructed from the SIP message object 114B, steps 2 and 3, and creates and  
5 edits the first line of the outgoing packet string object 110B, step 4, to contain the appropriate request type line. In steps 5 and 6, the first header objects 126C is read from the outgoing SIPMessage 114B and the appropriate line string is first created as a line string 154 and then appended to the outgoing packet string object 110 in steps 7 to 10. Steps 5 to 7 are repeated for all header objects in the  
10 vector list of the outgoing SIP message object 114B. Once the full string message 110B has been constructed, the Message Constructor object 152 passes the control message to the UserAgent, step 11.

It should be mentioned that not only are the header object classes dynamically loaded in this embodiment, but the UAThread class 46 is adapted to  
15 dynamically load the appropriate handler classes in step 10 of Figure 6, thus allowing ready extensibility of the request types handled by the program in the same manner as extensibility of header types handled by the program.

In the above-described embodiment with reference to Figures 6, 7 and 8, the header object classes are dynamically loaded from local storage. In a further  
20 embodiment, one or more header object classes, or handler object classes, may be remotely dynamically loaded by the RMI ClassLoader function or the URLClassLoader function. The program may identify the existence of an extension class for processing new request or header types by, for example a new header line received in an initial SIP message sent by a remote party, such as:  
25 New\_Header: address=132.146.107.43... or,  
Class\_Repository = <http://www.jungle.bt.co.uk/SIPheaders/>, as previously mentioned.

The header line may thus not only indicate the existence of the new class, but also identify the source from which the class may be obtained by  
30 dynamic loading.

It will be appreciated that further modifications and variations are also to be employed by the skilled person, without exceeding the scope of the present

invention. The invention is not only applicable to SIP, but also to other telecommunications protocols such as HTTP.

**Appendix A1 - Class.forName()**

```
public static Class forName (String className)
    throws ClassNotFoundException
```

5

Returns the Class object associated with the class or interface with the given string name. Invoking this method is equivalent to:

```
Class.forName(className, true, currentLoader)
```

10

Where currentLoader denotes the defining class loader of the current class.

For example, the following ocde fragment returns the runtime Class descriptor for the class named java.lang.Thread:

```
15      Class t = Class.forName("java.lang.Thread")
```

A call to forName("X") causes the class names X to be initialized.

**Appendix A2 – Class.forName()**

20

```
class MessageParser
{...
```

```
// "name" is the type of the header line.
```

```
25 // "line" is the String which contains the contents of that header line.
```

```
Class object = void.class;
```

```
Header h;
```

```
30  try{
```

```
    object = Class.forName(name);    //Load the class which is called "name"
```

```
    h = (Header)(object.newInstance());    //Create new object
```

```
    h.readString(line);    // The new "header" object reads /the line
```

```

        sip.addHeader(h);

    }
    catch(Exception e){
5      System.out.println("Class was not found");    //Part to change
    }

    ...}

```

# 10 Appendix A3

```

class MessageParser
{...

15    // "name" is the type of the header line.
    // "line" is the String which contains the contents of that header line.
    // "location" is the URL from which the remote class is available.

    Class object = void.class;

20    Header h;

    try{
        object = Class.forName(name);    //Load the class which is called "name"
        h = (Header)(object.newInstance());    //Create new object
25    h.readString(line);    // The new Header object reads the line

        sip.addHeader(h);    // Add this Header object to the vector
    in SIPMessage
    }

30    catch(Exception e){    //If class was not found locally
    then...
        URL urlList[] = { new URL(location)};

```

```

    ClassLoader loader;
    loader = new URLClassLoader(urlList);    // Create a URLClassLoader

    object = loader.loadClass (name); // Load the class which is called "name"
5
    h=(Header)object.newInstance();        // Create an instance of the class

    h.readString(line);                    // The new "header" object reads the line
    sip.addHeader(h);

10_ }

    ...}

```

#### Appendix B1 Header Interface

```

15
    public interface Header extends java.lang.Cloneable {
        Public String getTitle();           //Returns the title of the header it
                                           //represents.

        Public void readString(String line); //Reads the contents as a String and
20                                           //formulates any internal parameters.

        Public String getString();         //Returns the contents of the header in
                                           //a String format.

        Public String getType();           //Returns one of the type of the header I
                                           //it represents (as specified by the SIP
25                                           //RFC).

    }

```

#### Appendix B2 Example of a Header class

```

30 Public class call_id implements Header{
    private String id,title;
    public call_id(){                                //Public constructor
        id=new String("");

```

```

        title = new String("Call-ID");           //Initialise the title of the
                                                    //header
    }

5    public void readString(String s){id = s.trim();}
    public String getString(){return id;}
    public void setID(String s{id = s;}

                                                    //This is the only function added to the
                                                    //class other than those defined by the
10    //Header interface
    public String getType(){return "general-header";}

}

```

#### 15 Appendix B3 Part of MessageParser class

```

    public void makeSIPMessage(Vector v){ //When a SIP message arrived, each line
                                                    //was placed into a Vector as a
                                                    //separate //item. This vector is then
20    passed to //this function.

        .....
        for(int i=1;i<v.size();i++){
            String line = (String)v.elementAt(i); //Get the String representation of
                                                    //each header line

25    int index = line.indexOf(":");
        String name = (line.substring(0,index)).trim(); //Get the header's name
        name = name.replace('-', '_'); //Replace any '-' chars with '_'.
                                                    //This is necessary as a class
                                                    //name may not contain a '-'
30    //character.

        try{
            Class classname = void.class; //call corresponding class
            classname = Class.forName(name);

```

```
Header o;
try{
    O = (Header)(classname.newInstance());
}
5 catch(Exception e){
    System.out.println("Can not find class");
    Break;
}
o.readString((line.substring(index + 1)).trim()); //read the string.
10 sip.addHeader(o); //add to the sip message object.
}
catch(ClassNotFoundException e){System.out.println("Class was
not fund");}
catch(No ClassDefFoundError ee){System.out.println("Class was not
15 found");}
}
.....
}
```

## CLAIMS

- 5           1. A software application program for processing control messages constructed in accordance with a telecommunications control protocol, said program having an object-oriented structure and being arranged to dynamically load an object class, whereby a control message is processed, at run time.
- 10           2. A program according to claim 1, wherein said program is arranged to partition the contents of said control message and create a plurality of element objects from said control message, said objects being instances of classes dynamically loaded at run time.
- 15           3. A program according to claim 2, wherein said protocol defines a plurality of header types, said program being arranged to dynamically load a class or classes, representing each said header type, at run time.
4. A program according to claim 3, wherein said control message  
20 comprises a request line and a plurality of header lines, said plurality of header lines each containing a different respective header type.
5. A program according to claim 2, 3 or 4, wherein said program is arranged to create a message object containing said element objects as a vector  
25 array, and to pass said message object to a handler for processing in accordance with said protocol.
6. A program according to any preceding claim, wherein said protocol defines a plurality of control message types, said program being arranged to  
30 dynamically load a class or classes, for handling each said message type, at run time.

7. A program according to any preceding claim, wherein said program is arranged to dynamically load a class, whereby a control message, containing an element not specifically defined in said protocol, is processed.

5 8. A program according to any preceding claim, wherein said control message is an incoming message received from a remote party.

9. A program according to claim 8, wherein said dynamically loaded class or classes is/are loaded from said remote party.

10

10. A program according to any of claims 1 to 8, wherein said dynamically loaded class or classes is/are loaded from local storage.

11. A program according to any preceding claim, wherein said protocol  
15 is an application layer control protocol.

12. A program according to any preceding claim, wherein said protocol is a text-based protocol.

20 13. A software application program for processing control messages constructed in accordance with a telecommunications control protocol, said protocol defining a control message to have a construction including one or more protocol elements of a predefined format and having a plurality of respective types, wherein said program is arranged to:

25 process a control message by means of a different element type processing means for each of said plurality of respective types of at least one of said protocol elements; and

on receipt of a control message from a remote party that indicates that an element type processing means for a type of said at least one of said protocol  
30 elements is not available locally, to retrieve said unavailable element type processing means remotely in order to process a control message received from, or to be sent to, said remote party.

14. A program according to claim 13, wherein said program has an object-oriented structure, and said processing means is/are one or more object classes, and said retrieval comprises dynamically loading a class, which is unavailable locally, during run time.

5

15. A program according to claim 13 or 14, wherein said locally-unavailable element type processing means is retrieved from a source identified in a control message sent by said remote party.

10

16. A program according to any of claims 13 to 15, wherein said protocol is a session control protocol, and said program is arranged to detect the location of said locally-unavailable element type processing means, within a storage system at said remote party, in a control message received from said remote party.

15

17. A program according to claim 16, wherein said location is detected in a control message containing an invitation to join a session.

18. Data processing apparatus programmed with the software  
20 application program according to any preceding claim.

19. A computer workstation according to claim 18.

25

1/6

Fig.1.

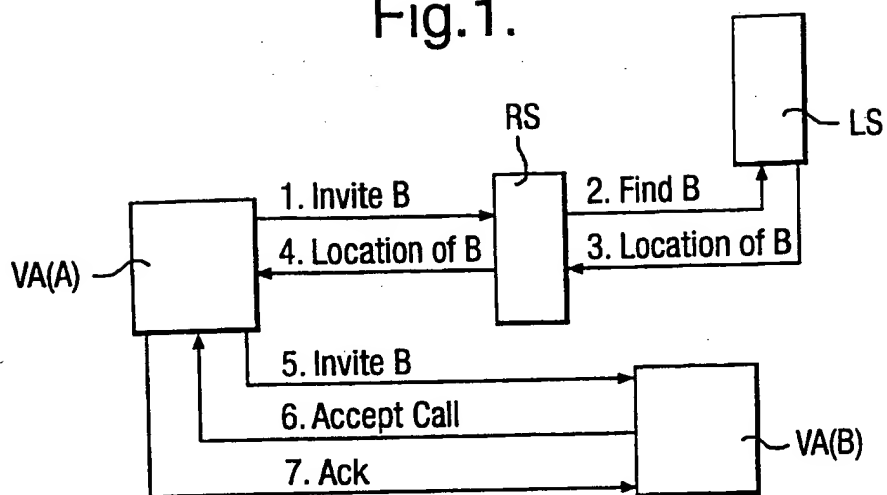
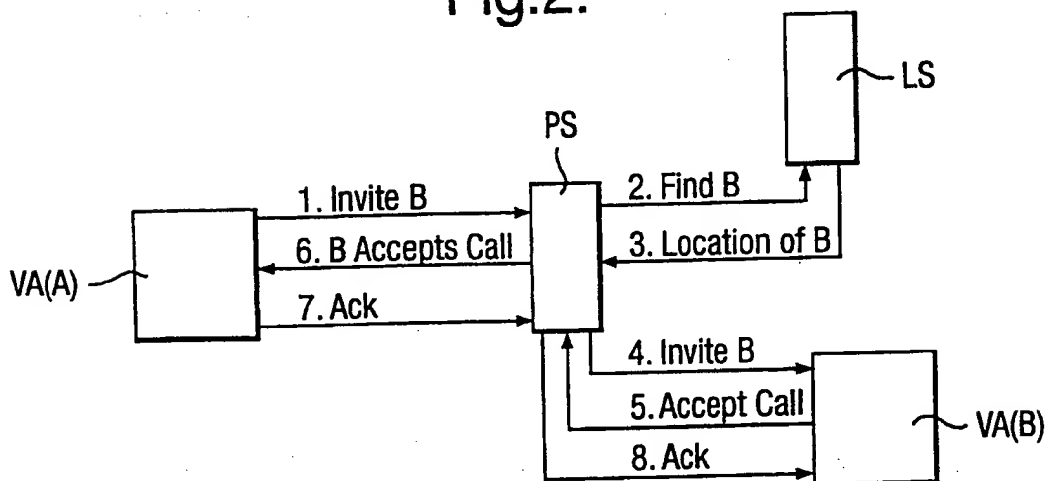


Fig.2.



2/6

Fig.3.

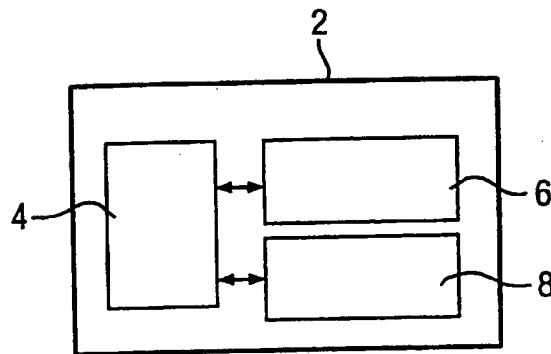
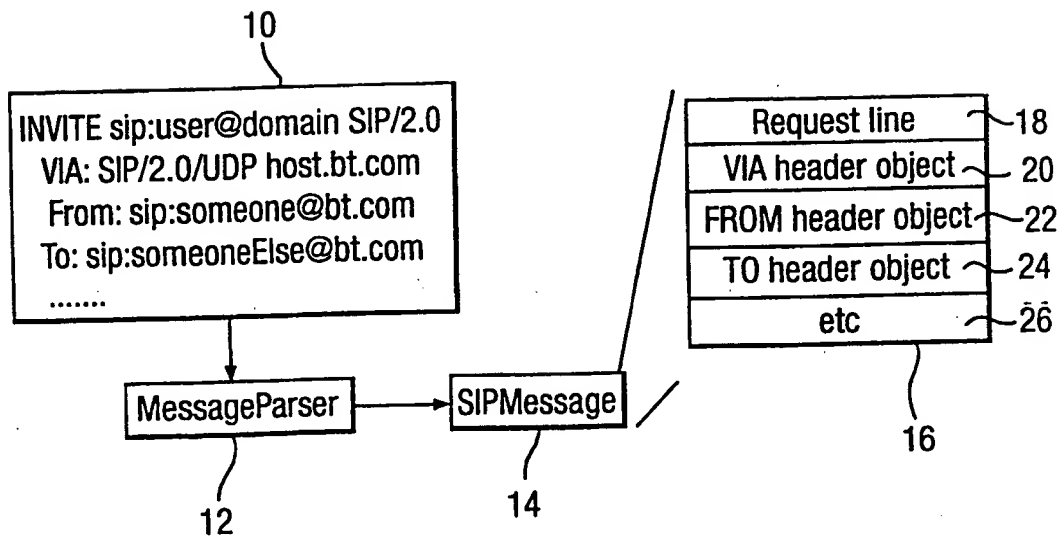


Fig.4.



3/6

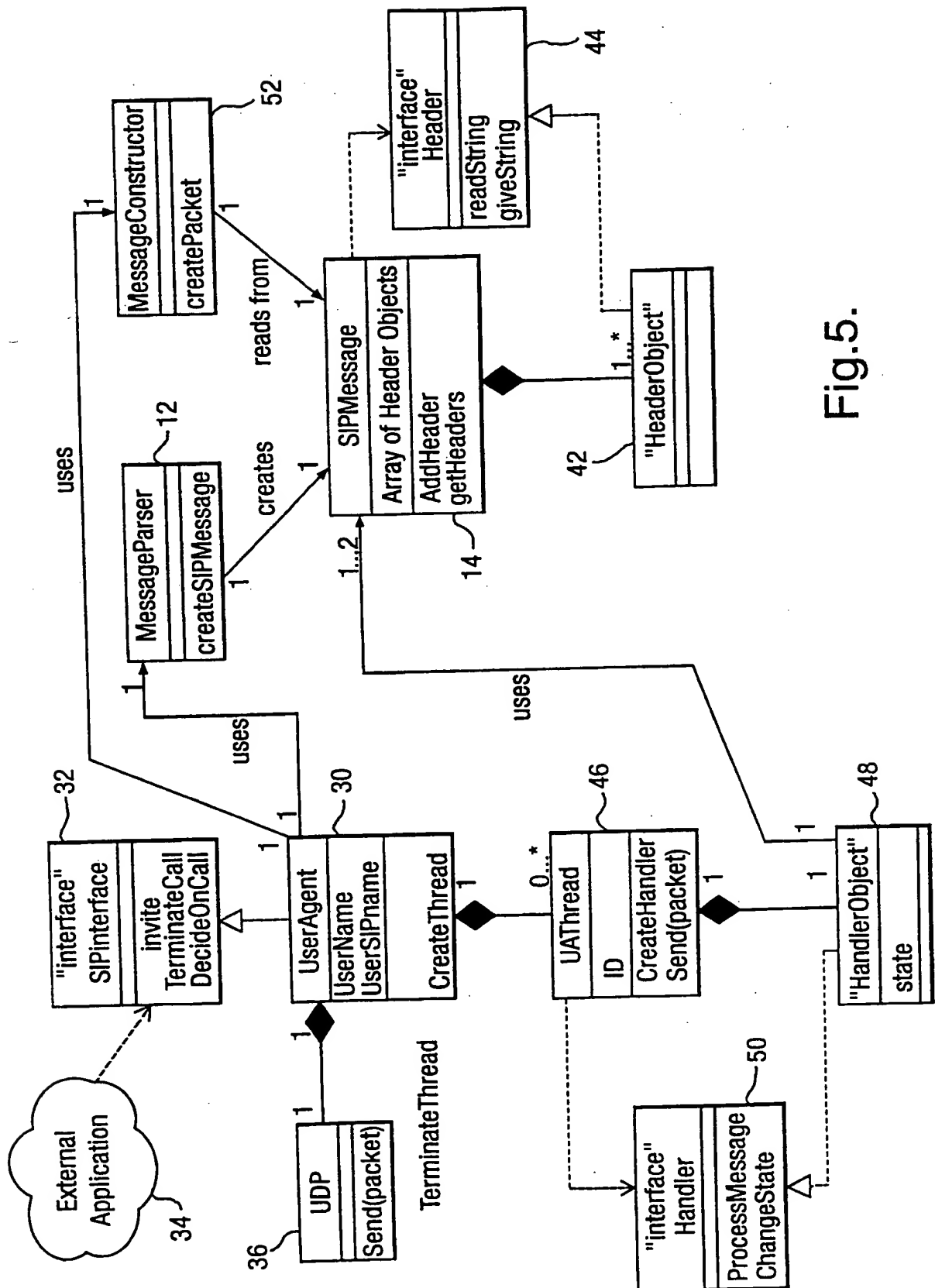


Fig.5.

4/6

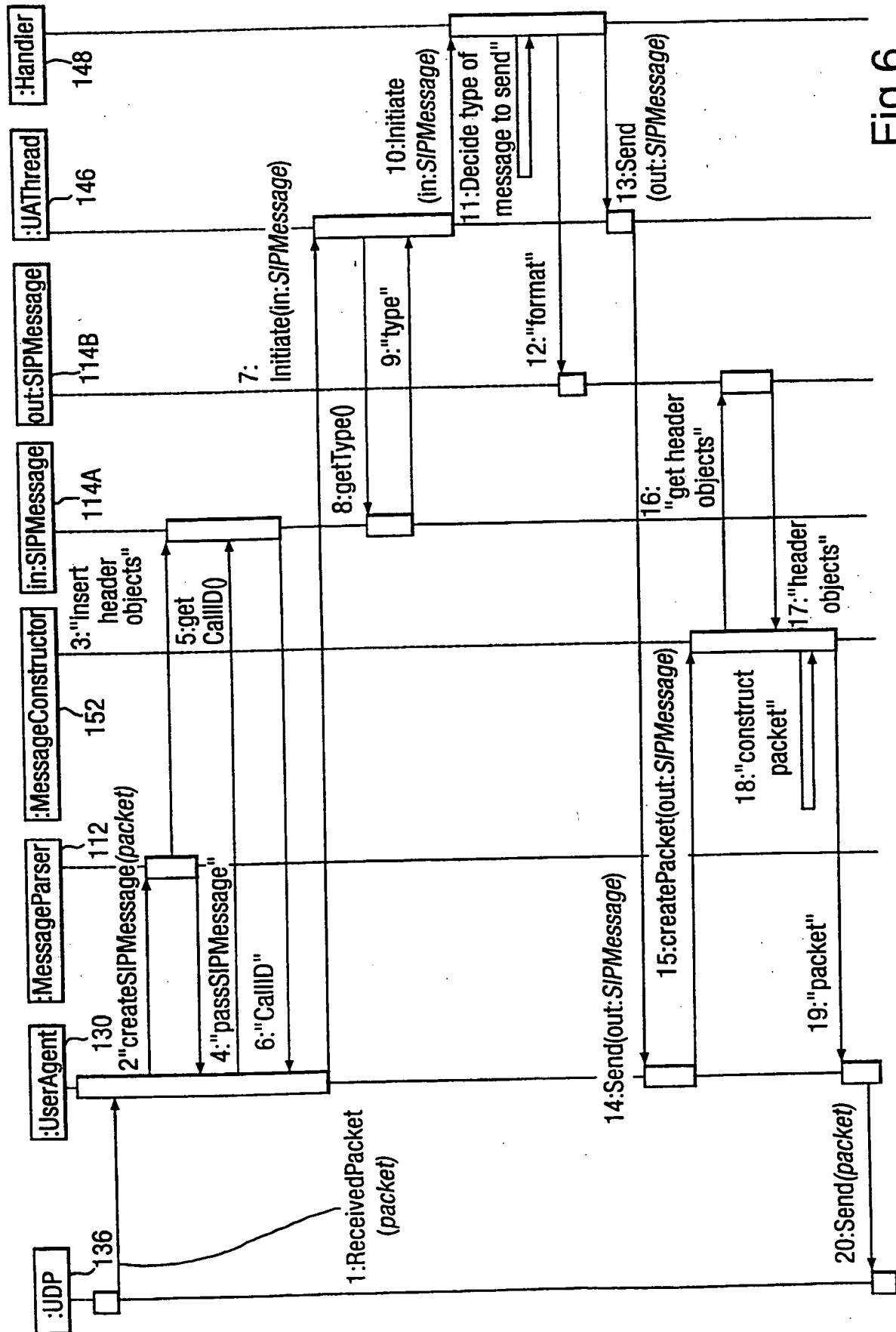


Fig.6.

Fig.7.

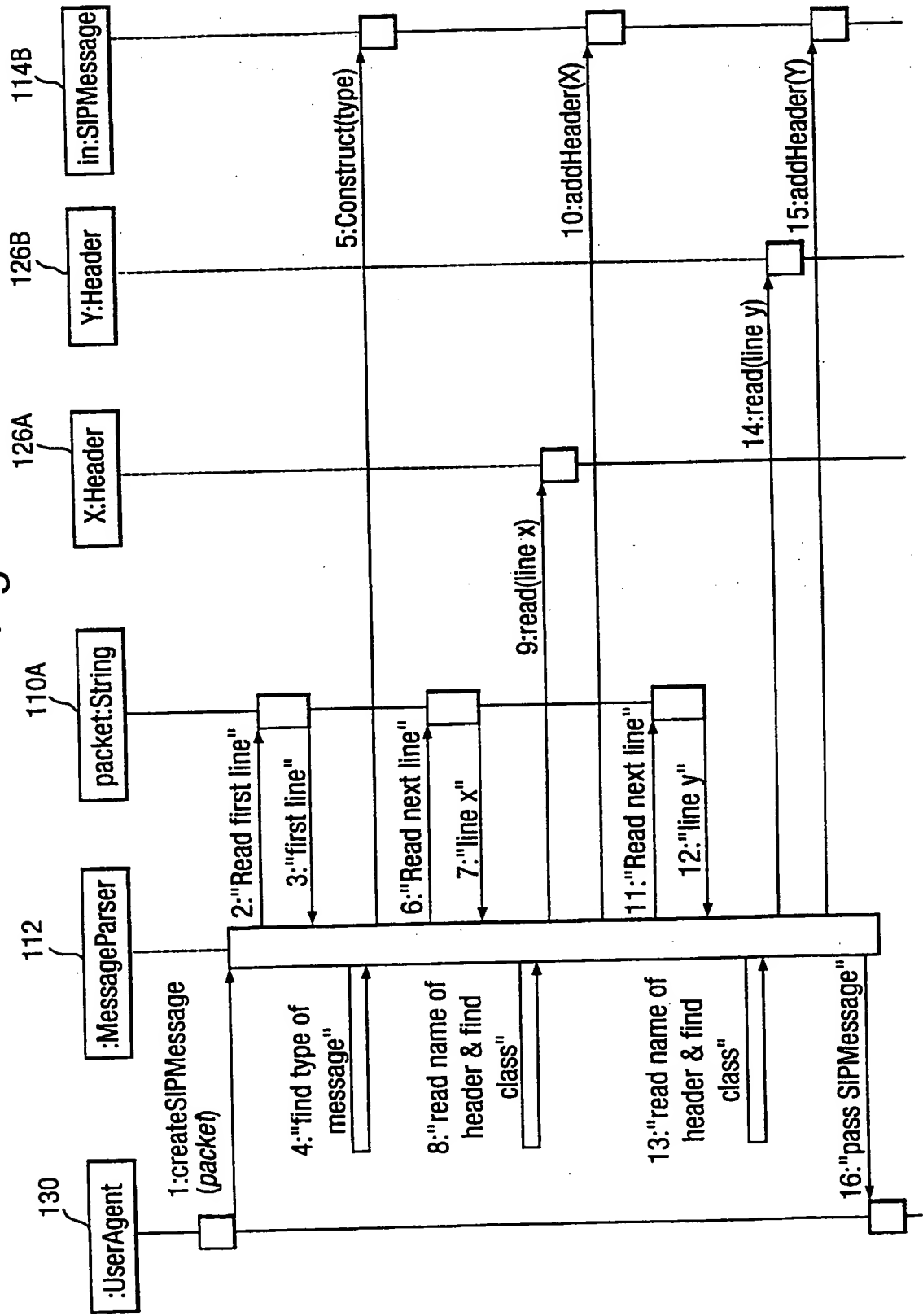
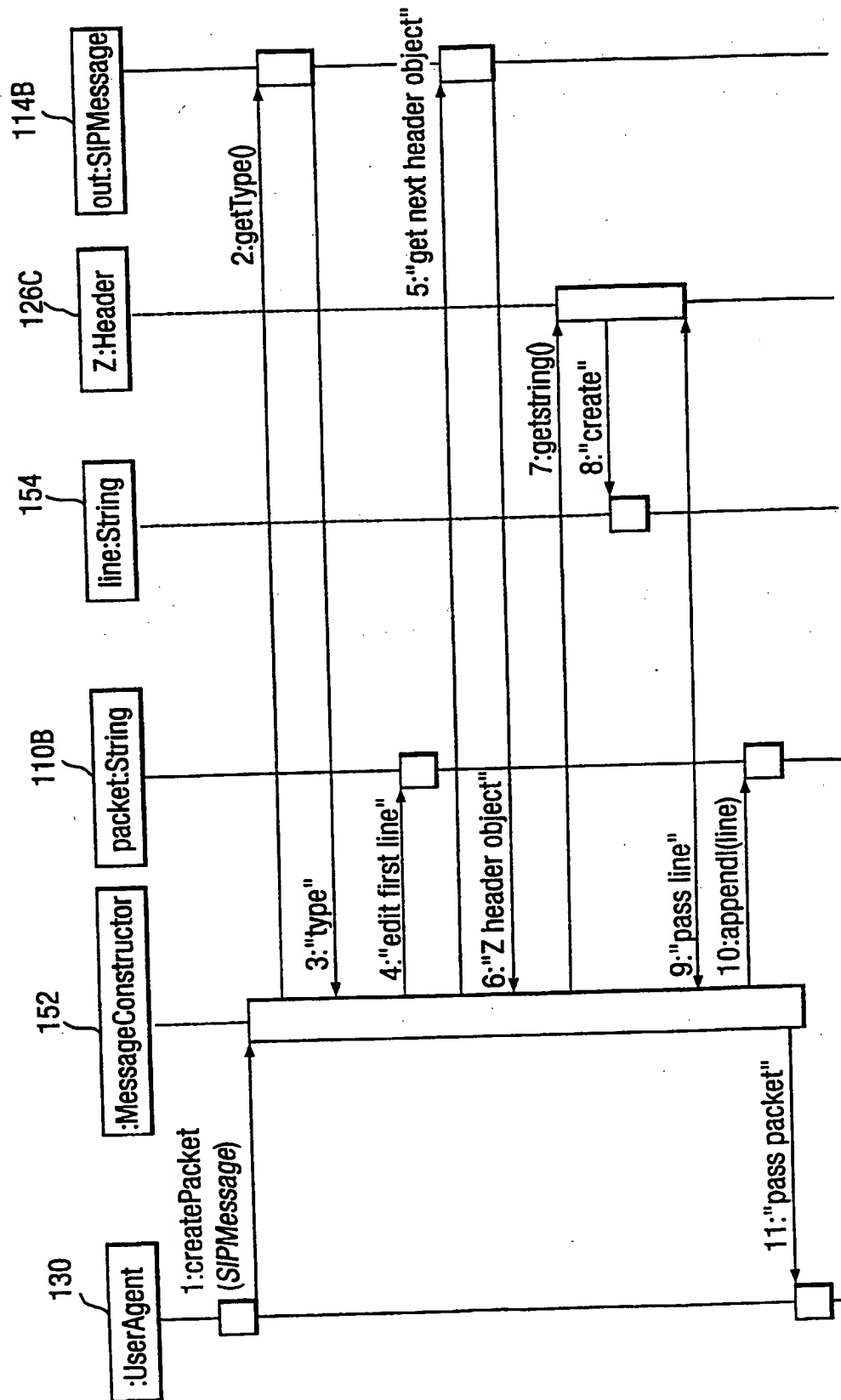


Fig.8.



**(19) World Intellectual Property Organization  
International Bureau**



**(43) International Publication Date**  
**17 May 2001 (17.05.2001)**

**(10) International Publication Number**  
**WO 01/35594 A3**

**PCT**

(51) **International Patent Classification<sup>7</sup>:** H04L 29/06

(21) **International Application Number:** PCT/GB00/04274

(22) **International Filing Date:**  
8 November 2000 (08.11.2000)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**  
99308869.9 8 November 1999 (08.11.1999) EP  
0000465.5 10 January 2000 (10.01.2000) GB

(71) **Applicant (for all designated States except US):** BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY [GB/GB]; 81 Newgate Street, London EC1A 7AJ (GB).

(72) **Inventor; and**

(75) **Inventor/Applicant (for US only):** GARYFALOS, Anargyros [GR/GB]; 10 Hunters Ride, Martlesham Heath, Ipswich, Suffolk IP5 3SQ (GB).

(74) **Agent:** ROBINSON, Simon, Benjamin; BT Group Legal Services, Intellectual Property Department, Holborn Centre, 8th floor, 120 Holborn, London EC1N 2TE (GB).

(81) **Designated States (national):** AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) **Designated States (regional):** ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published:**  
— with international search report

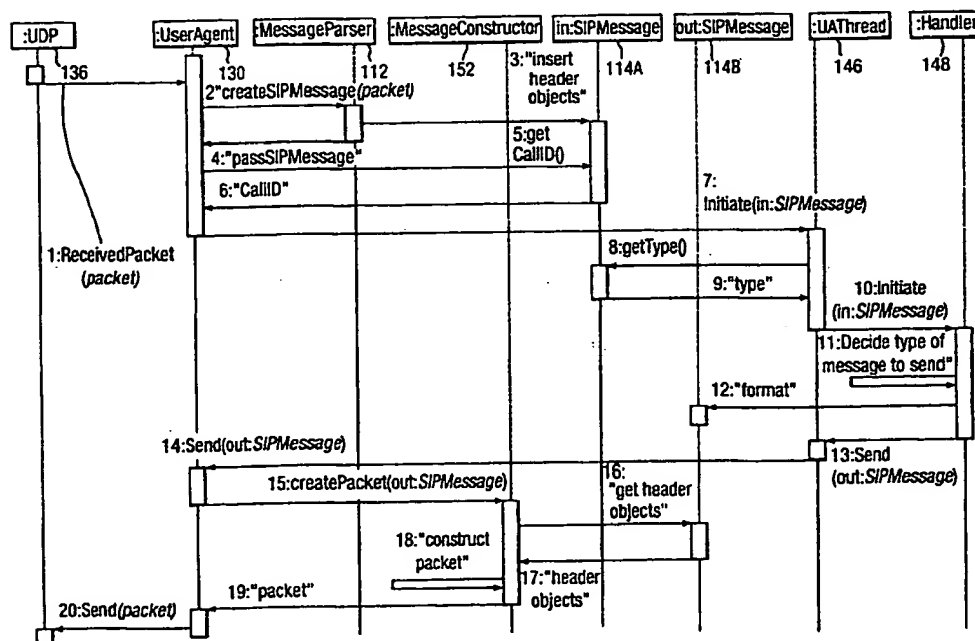
(88) **Date of publication of the international search report:**  
23 May 2002

**Published:**  
— *with international search report*

**(88) Date of publication of the international search report:**  
23 May 2002

*[Continued on next page]*

(54) Title: TELECOMMUNICATIONS CONTROL PROTOCOL PROCESSING



**(57) Abstract:** A software application program for processing control messages constructed in accordance with a telecommunications control protocol, said program having an object-oriented structure and being arranged to dynamically load an object class, whereby a control message is processed, at run time.

**WO 01/35594 A3**



*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

# INTERNATIONAL SEARCH REPORT

International Application No

PCT/GB 00/04274

## A. CLASSIFICATION OF SUBJECT MATTER

IPC 7 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC, IBM-TDB

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	<p>ANJUM, F.; CARUSO, F.; JAIN, R.; MISSIER, P.; ZORDAN, A.: "ChaiTime: a system for rapid creation of portable next-generation telephony services using third-party software components"</p> <p>OPEN ARCHITECTURES AND NETWORK PROGRAMMING PROCEEDINGS, 1999, 'Online!'</p> <p>26 - 27 March 1999, pages 22-31, XP002139285</p> <p>Openarch'99</p> <p>ISBN: 0-7803-5261-0</p> <p>Retrieved from the Internet:</p> <p>&lt;URL:www.iel.ihs.com&gt;</p> <p>'retrieved on 2000-05-31!'</p> <p>abstract</p> <p>page 23, left-hand column, line 11 - line 43</p> <p>page 27, right-hand column, line 10 - page 28, left-hand column, line 10</p> <p style="text-align: center;">-/--</p>	1,2,6-19



Further documents are listed in the continuation of box C.



Patent family members are listed in annex.

### \* Special categories of cited documents:

- \*A\* document defining the general state of the art which is not considered to be of particular relevance
- \*E\* earlier document but published on or after the international filing date
- \*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- \*O\* document referring to an oral disclosure, use, exhibition or other means
- \*P\* document published prior to the international filing date but later than the priority date claimed

- \*T\* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- \*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- \*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- \*G\* document member of the same patent family

Date of the actual completion of the international search

21 August 2001

Date of mailing of the international search report

28/08/2001

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Adkhis, F

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
28 June 2001 (28.06.2001)

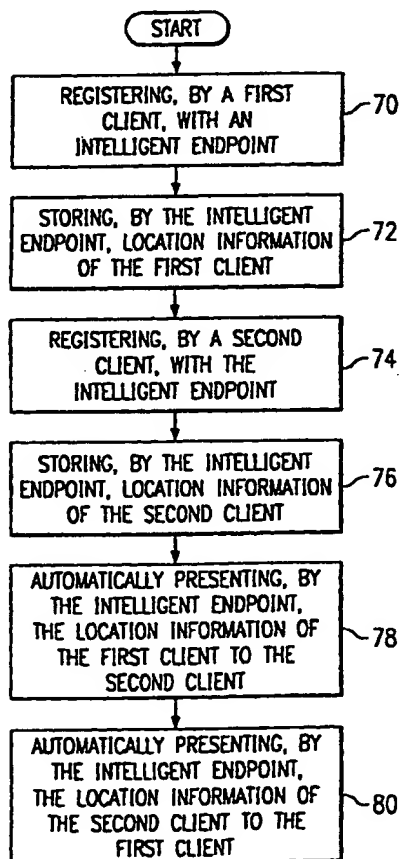
PCT

(10) International Publication Number  
**WO 01/47212 A1**

- (51) International Patent Classification<sup>7</sup>: H04L 29/06, 29/12
- (21) International Application Number: PCT/US00/34955
- (22) International Filing Date:  
20 December 2000 (20.12.2000)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:  
09/468,977 21 December 1999 (21.12.1999) US
- (71) Applicant (for all designated States except US): NORTEL NETWORKS LIMITED [CA/CA]; World Trade Center of Montreal, 8th Floor, 380 St. Antoine Street West, Montreal, Québec H2Y 3Y4 (CA).
- (72) Inventors; and  
(75) Inventors/Applicants (for US only): SOLLEE, Pat [US/US]; 2708 Garden Springs, Richardson, TX 75082 (US). JESSEN, Christopher [US/US]; 107 Ledgenest Drive, McKinney, TX 75070 (US).
- (74) Agents: MCCOMBS, David, L. et al.; Haynes and Boone, LLP, Suite 3100, 901 Main Street, Dallas, TX 75202 (US).
- (81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.
- (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European

[Continued on next page]

(54) Title: DISTRIBUTION OF LOCATION INFORMATION IN IP NETWORKS BY INTELLIGENT ENDPOINTS



(57) Abstract: A distribution of location information in IP networks by intelligent endpoints is presented. The network comprises an intelligent endpoint operably coupled to a plurality of clients. A first one of the clients registers with the intelligent endpoint that stores location information of the first client. A second one of the clients may also register with the intelligent endpoint that stores the location information of the second client. The intelligent endpoint automatically presents the location information of the first client to the second client and of the second client to the first client. The first client can then directly communicate with the second client. Either client can also be an intelligent endpoint.

WO 01/47212 A1



patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

— Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.

**Published:**

— With international search report.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

## DISTRIBUTION OF LOCATION INFORMATION IN IP NETWORKS BY INTELLIGENT ENDPOINTS

### Background

5        This application relates generally to distribution of location information in Internet Protocol (IP) networks and, more particularly, to distribution of location information in IP networks by intelligent endpoints.

10        With the location information, such as an IP address and/or name of a user, a client (or computer) may communicate with another client in the IP network. Such communication includes data, voice, and multi-media applications.

15        A conventional method for distributing location information in an IP network has been disclosed by the Internet Engineering Task Force (IETF) standards group with their Session Initiation Protocol (SIP). Fig. 1 depicts an IP network 10 that utilizes the SIP. The network 10 includes a SIP Proxy/Redirect Server 12 that receives and transmits messages between the SIP client's 14 (for example, the caller) and 16 (for example, the callee). The caller 14 and the callee 16 are 20 identified by their respective SIP addresses which are similar to an email address such as caller@host.com. In addition to the caller's 14 name, the caller's 14 telephone number could also be sent as callersnumber@host.com. The domain name ("host") can be a domain name or a numeric network address.

25        Before making a call, both the caller 14 and the callee 16 must Register 18 with the server 12. After registering and receiving an Acknowledge (not shown), the caller 14 can make a SIP call. To do so, the caller (SIP client) 14 sends an Invite 20 to the callee 16 via the server 12. The Invite 20 asks the callee 16 to join in a 30 communication session, such as a Voice over IP (VoIP) call, with the caller 14. The Invite 20 usually contains a session description that contains enough information for the callee 16 to join the communication. For multicast sessions, the session description numbers the types of media and formats that are permitted in the 35 communication session. For unicast sessions, the session description numbers the types of media and formats that the caller 14 is willing to use and to what location the information should be sent. If the callee 16 agrees to join the communication, the caller 14 is alerted

by receiving, from the callee 16, a similar description of the types of media and formats (in the case of a unicast transmission) that the callee 16 is willing to use. The caller 14 then confirms to the callee 16, via an Acknowledge (or Ack) message 22, that it 14 has  
5 received the callee's 16 confirmation. If the callee 16 wishes to end the communication, a Bye message (not shown) is sent to the caller 14.

There are certain limitations associated with this conventional method of distributing location information. For example, the caller 14 does not know if the callee 16 is available for the communication  
10 session. As such, network 10 time and capacity are constrained as messages are sent via the server 12 to initiate contact with a callee that may not be available. Further, the IP address of the callee 16 is not initially known by the caller 14 and thus degrades the network's 10 efficiency because of the messaging required to initiate  
15 a communication session. Additionally, the potential callers and callees do not receive an automatic update when a new client is available or unavailable for contact. Also, ownership, capacity, and privacy issues exist. The ownership issue deals with the responsibility for ensuring that the server 12 is functioning  
20 properly. If the server 12 goes "down," the users may not know the reason for the downed server or who to contact to get an update. The capacity issue deals with the fact that existing servers are free and thus often overloaded making communication difficult. The privacy issue deals with the fact that all users may "see" all other  
25 registered users and that the owner (or operator) of the server 12 can track users calls because they are being transported via the server 12.

In order to make an IP based call, the IP address of the called party must be known or determined before the call can be made. Today,  
30 users which dial into Internet Service Providers are allocated a different IP address each time they log in. Therefore, a method and network (such as an IP network) for distribution of location information, where a user (or username) is mapped to an IP address, that reduces or eliminates these limitations and inefficiencies is  
35 desired.

#### Summary

In response to these and other limitations, provided herein is a unique system and method for distributing location information in IP networks by intelligent endpoints.

The system (or IP network) of the present invention comprises an intelligent endpoint operably coupled to a plurality of clients. In one embodiment, a first one of the clients registers with the intelligent endpoint that stores location information of the first client. A second one of the clients may also register with the intelligent endpoint that stores the location information of the second client. The intelligent endpoint automatically presents the location information of the first client to the second client and of the second client to the first client.

In some embodiments the first client directly communicates with the second client.

In some embodiments the location information comprises the clients IP address.

In some embodiments the location information comprises a user's name, wherein the user is associated with one of the clients.

In some embodiments the intelligent endpoint comprises a static IP address.

In some embodiments the intelligent endpoint is associated with an area of interest.

In some embodiments the intelligent endpoint is associated with a plurality of areas of interest.

In some embodiments the intelligent endpoint registers with a second intelligent endpoint, where the second intelligent endpoint is associated with a second area of interest.

In some embodiments a plurality of clients register with a plurality of endpoints, where the registration includes each client's (or user's) areas of interest.

In some embodiments a private group of clients registers with the intelligent endpoint.

In some embodiments the first client de-registers with the intelligent endpoint which deletes the location information of the first client and automatically removes the location information of the first client with the second client.

These advantages, as well as others which will become apparent, are described in greater detail with respect to the drawings and the following disclosure.

## 5    **Brief Description of the Drawings**

Fig. 1 is a diagrammatic view of a prior art communication system for distributing location information.

Fig. 2 is a diagrammatic view of a communication system of the present invention for distributing location information.

10    Fig. 3 is a computer of the present invention.

Fig. 4 is a flow chart of a method for distributing location information in an Internet Protocol network of the present invention.

## **Detailed Description**

Fig. 2 depicts an Internet Protocol (IP) network 28 of the present invention. The network 28 includes an intelligent endpoint (or super client) 30 that is operably coupled to a plurality of clients 32-38. The intelligent endpoint 30, which may be SIP based (i.e. an extension of SIP) comprises a static IP address and includes an always available IP connection (via, for example, a cable modem or Digital Subscriber Line). The user of the intelligent endpoint 30 provides its IP address to the plurality of clients 32-38. If client 32 (Joe) decides to go on-line, he registers, via a Register message 40 with the intelligent endpoint 30 that stores the location information of the client 32. The Register message 40 includes the location information (such as an IP address or user's name - where the user is associated with one of the clients) of the client 32. After the client 32 has registered, it 32 receives a message from the intelligent endpoint 30 indicating that it is the only client registered thus far. As such, Joe may want to initiate a communication with Mary.

If client 34 (Pat) then registers, its 34 location information is stored with the intelligent endpoint 30. The intelligent endpoint 30 then automatically sends a message to the client 34 indicating (or presenting) to Pat that Joe is also on-line and provides the client's 32 location information. Similarly, the intelligent endpoint 30 automatically sends a message to the client 32 indicating to Joe that Pat is on-line and provides the client's 34 location information.

The clients location information may be presented via a "pop-up" window, such as a Graphical User Interface, that can be accessed (for example, "double clicked") to initiate a communication session. The communication may be, for example, a VoIP call, a multi-media  
5 conference, or a "text chat" call. Either client 32, 34 may now directly communicate with the other (or with the intelligent endpoint 30).

As each additional client 36, 38 registers with the intelligent endpoint 30, they 36, 38, as well as the clients 32, 34, are updated  
10 with each others location information and thus their users (Joe, Pat, Fred, and Bob) can easily communicate with one another. As such, after the clients have registered with the intelligent endpoint 30, no messaging must traverse the intelligent endpoint 30 for communication between the clients.

15 If a client wishes to go off-line (or de-register) a Register message, with a time out value of zero (not shown), is sent to the intelligent endpoint 30. The intelligent endpoint 30 deletes the location information of the client and automatically removes that information from the remaining clients'  
20 pop-up window (for example) that are still on-line. Thus, the remaining clients are immediately updated of this fact and know to not attempt a communication with the departed client.

Intelligent endpoints of the present invention are associated with an area of interest (or group). For example, the intelligent  
25 endpoint 30 may be associated with the "Dallas Cowboys" area of interest. The users are probably interested in communicating with other Dallas Cowboy fans and thus register with the intelligent endpoint 30. The intelligent endpoint 30 may also be associated with many areas of interest (or groups) and may register (by sending its IP  
30 address) with another intelligent endpoint 50 that is associated with another area of interest, such as "High-Tech Stocks." The clients 52, 54 of the intelligent endpoint 50 may communicate with one another in a similar manner as described above. Additionally, the clients 52, 54 may register with the intelligent endpoint 30 and communicate with  
35 clients 32-38. As such, Julie may communicate directly with Bob.

In addition to including a client's location information, each registration message may also include a client's (or user's) areas of

interest. As such, if an intelligent endpoint registers with another intelligent endpoint that includes an area of interest specified by a user, that user's client will be alerted to that fact allowing the user to communicate with other users who have the same interest.

- 5 Additionally, users may set up private groups by only allowing pre-defined users' clients to register with an intelligent endpoint.

Any of the clients 32-38, 52, 54 may become intelligent endpoints if they can provide a static IP address and an always available IP connection. As such, Pat may decide to start a new group and could  
10 alert others that his client 34 is now an intelligent endpoint for a new area of interest. Those clients who are interested in the new area, may register with Pat's client 34 and communicate with one another in a similar manner as described above. As this communication is occurring, Pat may simultaneously communicate with other clients  
15 and/or other intelligent endpoints.

Fig. 3 depicts a computer 60 (which contains a computer program) that comprises a processor 62 and memory 64. The computer 60 may be a personal computer or laptop, the clients 32-38, 52-54, the intelligent endpoints 30, 50, and/or any device that can send and receive IP  
20 information. The processor 62 may be a central processing unit, digital signal processor, microprocessor, microcontroller, microcomputer, and/or any device that manipulates digital information based on programming instructions. The memory 64 may be read-only memory, random access memory, flash memory and/or any device that  
25 stores digital information. The memory 64 is coupled to the processor 62 and stores programming instructions (the computer program) that, when read by the processor 62, cause the processor to perform certain processing operations.

Fig. 4 depicts a method for distributing location information in  
30 an IP network that may be implemented by the computers described above. The method begins at step 70 where a first client registers with an intelligent endpoint that stores, at step 72, location information of the first client. At step 74, a second client registers with the intelligent endpoint that stores, at step 76,  
35 location information of the second client. The method proceeds to step 78 where the intelligent endpoint automatically presents the location information of the first client to the second client and, at

step 80, automatically presents the location information of the second client to the first client

The present invention thus enjoys several advantages. For example, an intelligent endpoint is described that efficiently and effectively distributes location information, which includes a client's IP address and/or a user name, across an IP network. Additionally, a user may specify, to the intelligent endpoint, the groups or area's he/she is interested in. Also, the ownership, capacity and privacy issues of the prior art are addressed because the intelligent endpoint contact is known to the users, the size of the groups are generally small, and the users only "see" other registered users in the same intelligent endpoint and/or in another intelligent endpoint registered with the same intelligent endpoint. Further, the improvements of the present invention may be applied to and used with the SIP, SIP Proxy/Redirect Server, and the SIP clients.

It is understood that variations may be made in the foregoing without departing from the scope of the present invention. For example, any number and combination of entities such the clients 32-38, 52-54, the intelligent endpoints 30, 50, and the computer 60, may comprise or be used with the present network 28. Also, the network 28 may be connected to another wireless, wireline, data, voice, and/or multi-media network. Further, other parameters may be included in the Register message 40 that provides the intelligent endpoint 30 with additional information. For example, a parameter may inform the intelligent endpoint 30 that a client is an intelligent endpoint enabled client and thus has certain privileges and rights.

It is further understood that other modifications, changes and substitutions are intended in the foregoing disclosure and in some instances some features of the disclosure will be employed without corresponding use of other features. Additionally, singular discussion of items and/or computers located in the network 28 is also meant to apply to situations where a plurality of items and/or computers exist. Accordingly, it is appropriate that the appended claims be construed broadly and in a manner consistent with the scope of the disclosure.

## WHAT IS CLAIMED IS:

1. A method for distributing location information in an Internet Protocol (IP) network that comprises an intelligent endpoint operably coupled to clients, the method comprising:

5 registering, by a first one of the clients, with the intelligent endpoint;

storing, by the intelligent endpoint, the location information of the first client;

10 registering, by a second one of the clients, with the intelligent endpoint;

storing, by the intelligent endpoint, the location information of the second client;

automatically presenting, by the intelligent endpoint, the location information of the first client to the second client; and

15 automatically presenting, by the intelligent endpoint, the location information of the second client to the first client.

2. The method of claim 1 further comprising directly communicating, by the first client, with the second client.

20

3. The method of claim 1, wherein the location information comprises the clients IP address.

4. The method of claim 1, wherein the location information comprises a user's name, wherein the user is associated with one of the clients.

25

5. The method of claim 1, wherein the intelligent endpoint comprises a static IP address.

30 6. The method of claim 1, wherein the intelligent endpoint is associated with an area of interest.

7. The method of claim 1, wherein the intelligent endpoint is associated with a plurality of areas of interest.

35 8. The method of claim 1 further comprising registering by the intelligent endpoint with a second intelligent endpoint, wherein the

second intelligent endpoint is associated with a second area of interest.

9. The method of claim 1 further comprising registering, by a  
5 plurality of clients, with a plurality of intelligent endpoints,  
wherein the registration includes each client's areas of interest.

10. The method of claim 1 further comprising registering, by a  
private group of clients, with the intelligent endpoint.

10

11. The method of claim 1 further comprising:

de-registering, by the first client, with the intelligent  
endpoint;

15 deleting, by the intelligent endpoint, the location information  
of the first client; and

automatically removing, by the intelligent endpoint, the location  
information of the first client with the second client.

12. A system for distributing location information in an Internet  
20 Protocol (IP) network that comprises an intelligent endpoint operably  
coupled to clients, the system comprises:

means for registering, by a first one of the clients, with the  
intelligent endpoint;

25 means for storing, by the intelligent endpoint, the location  
information of the first client;

means for registering, by a second one of the clients, with the  
intelligent endpoint;

means for storing, by the intelligent endpoint, the location  
information of the second client;

30 means for automatically presenting, by the intelligent endpoint,  
the location information of the first client to the second client; and

means for automatically presenting, by the intelligent endpoint,  
the location information of the second client to the first client.

35 13. The system of claim 1 further comprises means for directly  
communicating, by the first client, with the second client.

14. The system of claim 1, wherein the location information comprises the clients IP address.

15. The system of claim 1, wherein the location information comprises  
5 a user's name, wherein the user is associated with one of the clients.

16. The system of claim 1, wherein the intelligent endpoint comprises a static IP address.

10 17. The system of claim 1, wherein the intelligent endpoint is associated with an area of interest.

18. The system of claim 1, wherein the intelligent endpoint is associated with a plurality of areas of interest.

15

19. The system of claim 1 further comprises means for registering by the intelligent endpoint with a second intelligent endpoint, wherein the second intelligent endpoint is associated with a second area of interest.

20

20. The system of claim 1 further comprises means for registering, by a plurality of clients, with a plurality of endpoints, wherein the registration includes each client's areas of interest.

25 21. The system of claim 1 further comprises means for registering, by a private group of clients, with the intelligent endpoint.

22. The system of claim 1 further comprises:  
means for de-registering, by the first client, with the  
30 intelligent endpoint;

means for deleting, by the intelligent endpoint, the location information of the first client; and

means for automatically removing, by the intelligent endpoint, the location information of the first client with the second client.

35

23. An intelligent endpoint for distributing location information in an Internet Protocol (IP) network, wherein the IP network includes

clients operably coupled to the intelligent endpoint, the intelligent endpoint comprises:

- means for registering a first one of the clients;
- means for storing the location information of the first client;
- 5 means for registering a second one of the clients;
- means for storing the location information of the second client;
- means for automatically presenting the location information of the first client to the second client; and
- means for automatically presenting the location information of the second client to the first client.

24. An Internet Protocol (IP) network comprises:
- an intelligent endpoint;
  - clients operably coupled to the intelligent endpoint, wherein the
  - 15 intelligent endpoint performs at least one of the following steps from a group consisting of:
    - registering a first one of the clients;
    - storing location information of the first client;
    - registering a second one of the clients;
    - 20 storing location information of the second client;
    - automatically presenting the location information of the first client to the second client; and
    - automatically presenting the location information of the second client to the first client.

25

25. A computer program comprising instructions for:
- registering, by a first client, with an intelligent endpoint, wherein the first client is operably coupled to the intelligent endpoint;
  - 30 storing, by the intelligent endpoint, location information of the first client;
  - registering, by a second one of the clients, with the intelligent endpoint wherein the second client is operably coupled to the intelligent endpoint;
  - 35 storing, by the intelligent endpoint, location information of the second client;

automatically presenting, by the intelligent endpoint, the location information of the first client to the second client; and automatically presenting, by the intelligent endpoint, the location information of the second client to the first client.

5

26. The computer program of claim 25 further comprising instructions for directly communicating, by the first client, with the second client.

10

27. The computer program of claim 25, wherein the location information comprises the clients IP address.

15

28. The computer program of claim 25, wherein the location information comprises a user's name, wherein the user is associated with one of the clients.

29. The computer program of claim 25, wherein the intelligent endpoint comprises a static IP address.

20

30. The computer program of claim 25, wherein the intelligent endpoint is associated with an area of interest.

31. The computer program of claim 25, wherein the intelligent endpoint is associated with a plurality of areas of interest.

25

32. The computer program of claim 25 further comprising instructions for registering by the intelligent endpoint with a second intelligent endpoint, wherein the second intelligent endpoint is associated with a second area of interest.

30

33. The computer program of claim 25 further comprising instructions for registering, by a plurality of clients, with a plurality of endpoints.

35

34. The computer program of claim 25 further comprising instructions for registering, by a private group of clients, with the intelligent endpoint.

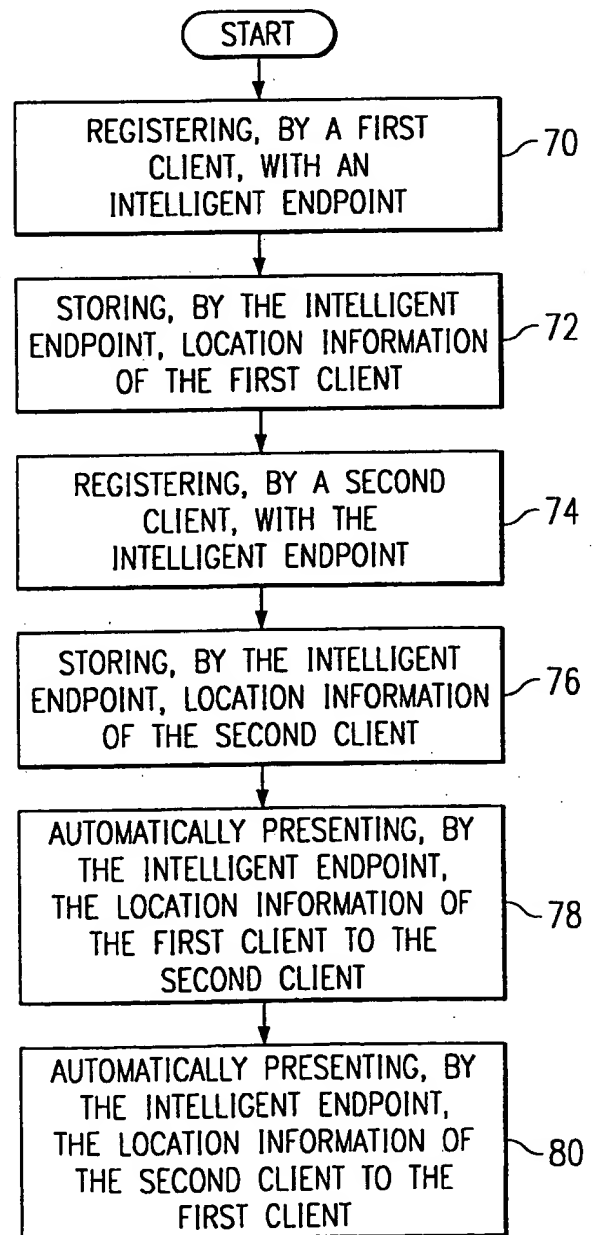
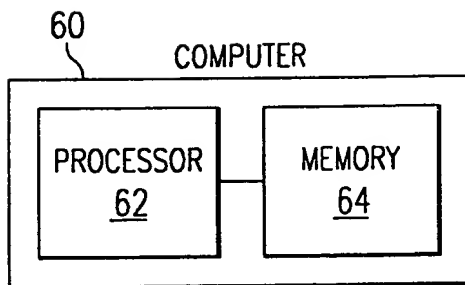
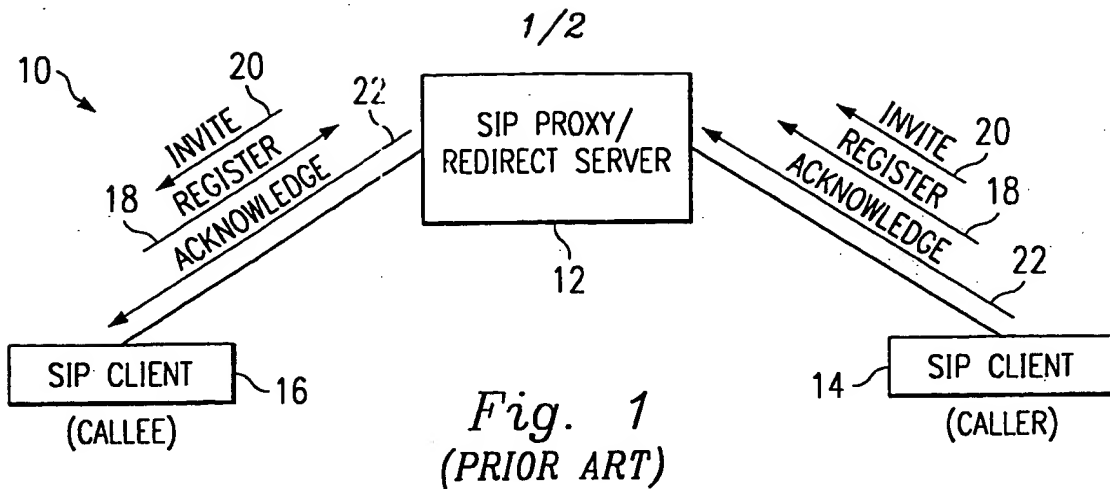
35. The computer program of claim 25 further comprising instructions for:

5 de-registering, by the first client, with the intelligent endpoint;

deleting, by the intelligent endpoint, the location information of the first client; and

automatically removing, by the intelligent endpoint, the location information of the first client with the second client.

10



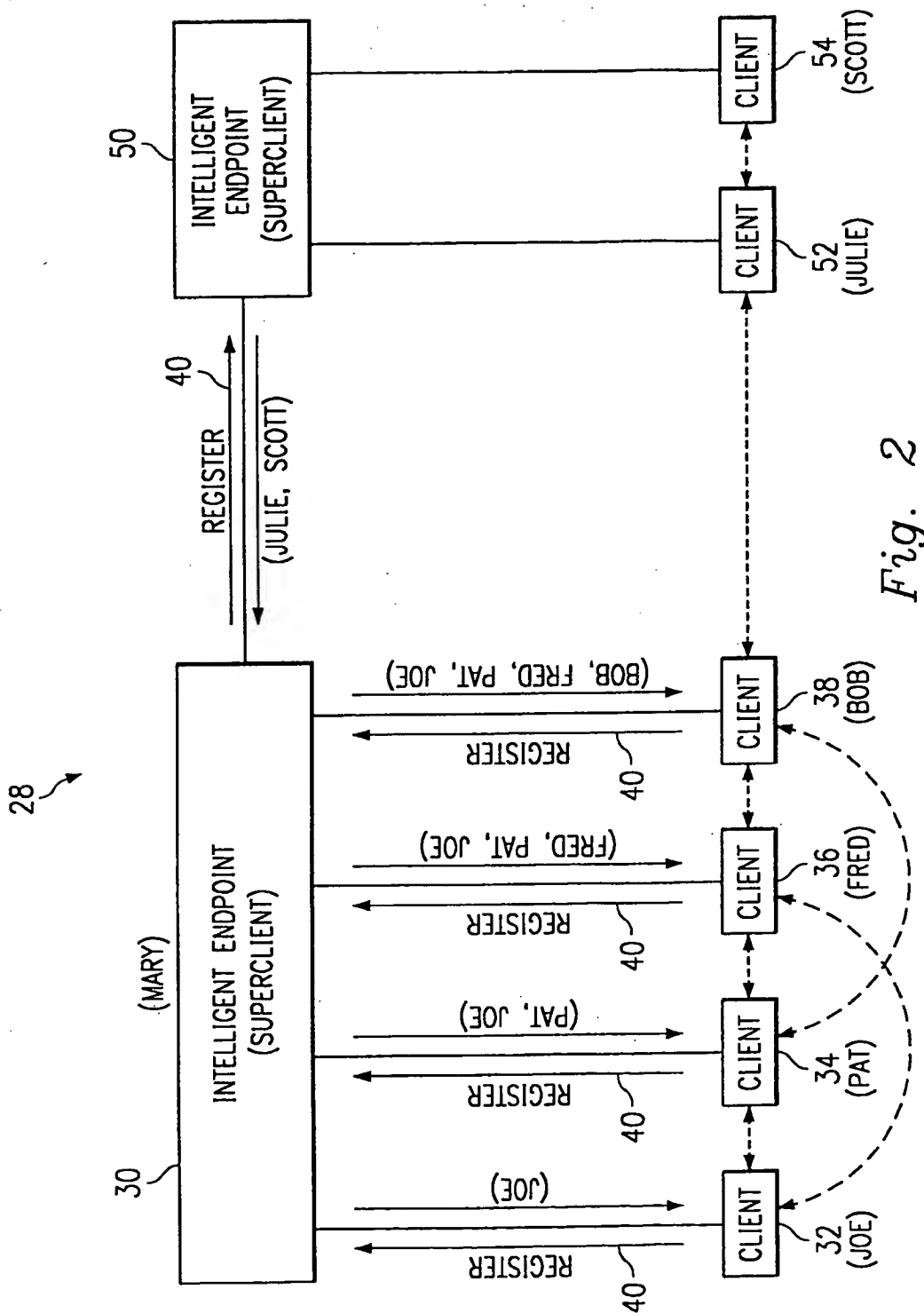


Fig. 2

# INTERNATIONAL SEARCH REPORT

International Application No

PCT/US 00/34955

**A. CLASSIFICATION OF SUBJECT MATTER**  
IPC 7 H04L29/06 H04L29/12

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the International search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 98 16045 A (MIRABILIS LTD ;VARDI ARIEH (IL); AMNON AMIR (IL); GOLDFINGER YAIR) 16 April 1998 (1998-04-16)	1-5, 10-16, 21-29, 33-35
Y	abstract; claim 1; figure 1  page 2, line 15 - line 34 page 3, line 11 - line 28 page 7, line 26 -page 8, line 14 page 9, line 27 - line 30 page 10, line 16 - line 25 page 11, line 22 - line 29 --- -/--	6-9, 17-20, 30-32

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

\* Special categories of cited documents:

- \*A\* document defining the general state of the art which is not considered to be of particular relevance
- \*E\* earlier document but published on or after the International filing date
- \*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- \*O\* document referring to an oral disclosure, use, exhibition or other means
- \*P\* document published prior to the international filing date but later than the priority date claimed

- \*T\* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- \*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- \*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- \* & \* document member of the same patent family

Date of the actual completion of the international search

14 May 2001

Date of mailing of the international search report

21/05/2001

Name and mailing address of the ISA  
European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Stergiou, C

# INTERNATIONAL SEARCH REPORT

International Application No

PCT/US 00/34955

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	<p>MULLER N: "DIAL 1-800-INTERNET WITH NEW SOFTWARE, YOU CAN TALK BUSINESS OVER THE NET AND AVOID LONG-DISTANCE CHARGES" BYTE,US,MCGRRAW-HILL INC. ST PETERBOROUGH, vol. 21, no. 2, 1 February 1996 (1996-02-01), pages 83-84,86,88, XP000566097 ISSN: 0360-5280 page 83, column 1, line 1, paragraph 1 -page 84, column 3, paragraph 3; figure 1</p>	<p>6-9, 17-20, 30-32</p>
A	<p>TOGA J ET AL: "ITU-T STANDARDIZATION ACTIVITIES FOR INTERACTIVE MULTIMEDIA COMMUNICATIONS ON PACKET-BASED NETWORKS: H.323 AND RELATED RECOMMENDATIONS" COMPUTER NETWORKS AND ISDN SYSTEMS,NORTH HOLLAND PUBLISHING. AMSTERDAM,NL, vol. 31, no. 3, 11 February 1999 (1999-02-11), pages 205-223, XP000700319 ISSN: 0169-7552 page 210, column 1, paragraph 3.4 -page 211, column 1, paragraph 4</p>	<p>1-35</p>

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/US 00/34955

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9816045 A	16-04-1998	AU 4395497 A	05-05-1998
		BR 9712207 A	25-01-2000
		CN 1240083 A	29-12-1999
		EP 1013044 A	28-06-2000
<hr/>			